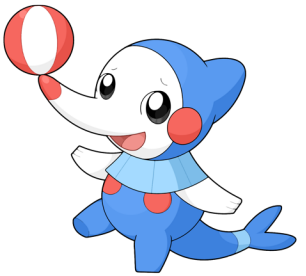
数值计算之美

SHU ZHI JI SUAN ZHI MEI

胡家威

http://hujiaweibujidao.github.io/



清华大学逸夫图书馆 · 北京

内 容 简 介

本书是我对数值计算中的若干常见的重要算法及其应用的总结，内容还算比较完整。

本人才疏学浅，再加上时间和精力有限，所以本书不会详细介绍很多的概念，需要读者有一定的基础或者有其他的参考书籍，这里推荐参考文献中的几本关于数值计算的教材。

本书只会简单介绍下算法的原理，对于每个算法都会附上我阅读过的较好的参考资料以及算法的实现 (Matlab 或者其他语言)，大部分代码是来源于参考文献 [1] 或者是经过我改编而成的，肯定都是可以直接使用的，需要注意的是由于 Latex 对代码的排版问题，导致中文注释中的英文字符经常出现错位，对于这种情况请读者自行分析，不便之处还望谅解。写下这些内容的目的是让自己理解地更加深刻些，顺便能够作为自己的 HandBook，如有错误之处，还请您指正，本人邮箱地址是：hujiawei090807@gmail.com。

目　　录

第四章　曲线拟合和多项式插值

曲线拟合 (Curve Fitting) 是使用一个数学表达式去拟合一些数据点，目的是找到一个最好的能够拟合这些数据点的表达式。而插值 (Interpolation) 不同，插值是首先确定一个能够完全通过数据点的多项式表达式，然后对数据点之间的点进行准确值估计的过程。

4.1　曲线拟合

4.1.1　使用线性方程进行曲线拟合

使用线性方程 (一阶多项式) 进行曲线拟合就是使用 $y = ax + b$ 形式来拟合曲线，只要确定常量 a 和 b 即可。关键是怎么衡量是一个比较好的拟合呢？最常用的是最小二乘法，也就是使得准确值与拟合值之差的平方和达到最小，例如对于一些数据点 (x_i, y_i)，全局误差 E 为：

$$E = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

要保证 E 最小，那么 E 对 a 和 b 的偏导数为 0。假设有下面四组求和：

$$S_(x) = \sum_{i=1}^{n} x_i \ , S_(y) = \sum_{i=1}^{n} y_i \ , S_(xy) = \sum_{i=1}^{n} x_i y_i \ , S_(xx) = \sum_{i=1}^{n} x_i^2$$

那么可以得到常数 a 和 b 分别为：

$$a = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x^2)} \qquad b = \frac{S_{xx}S_y - S_{xy}S_x}{nS_{xx} - (S_x^2)}$$

使用线性方程进行曲线拟合的 Matlab 代码实现：

code/linearcfnm.m

```
function [a,b] = linearcfnm(x,y)
%使用线性方程进行曲线拟合
% t=0:10:100;
% p=[0.94 0.96 1.0 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28];
% [a b]=linearcfnm(t,p)

n=length(x);
Sx=sum(x);
Sy=sum(y);
Sxy=sum(x.*y);
Sxx=sum(x.^2);
a=(n*Sxy-Sx*Sy)/(n*Sxx-Sx^2);
b=(Sy*Sxx-Sx*Sxy)/(n*Sxx-Sx^2);
end
```

4.1.2　非线性方程进行曲线拟合

很多情况下，数据点并不能够通过线性方程进行拟合，而是需要使用非线性方程进行拟合。然而很多的非线性方程的拟合问题可以转换成线性方程拟合的问题求解，常见的有以下几种：

(1)$y = bx_m$ -> $ln(y) = mln(x) + ln(b)$

(2)$y = be^{mx}$ -> $ln(y) = mx + ln(b)$

(3)$y = \frac{1}{mx+b}$ -> $\frac{1}{y} = mx + b$

(4)$y = \frac{mx}{b+x}$ -> $\frac{1}{y} = \frac{b}{m}\frac{1}{x} + \frac{1}{m}$

4.1.3 使用二次或者高次多项式进行曲线拟合 [最小二乘问题]

对于包含 n 个数据点的曲线拟合问题，只有阶数为 $(n-1)$ 的多项式才可以完整通过所有的点，但是对于高阶多项式，虽然它在数据点处得到准确解，在内部或者外部某些点进行插值问题时得到的解也有可能不可靠。详细的例子可以参考文献 [3](P166-P167)。

使用高次多项式进行曲线拟合基本思想是用 n 个基函数的线性组合来近似 y。这个时候我们得到一个方程组 $X\beta \approx y$，其中 X 为设计矩阵，大小为 $(m*n)$，m 是数据点的个数，n 是多项式的阶数，也就是需要确定的参数的个数。一般情况下，这类问题都是 $m > n$，所以这是一个超定方程组，没有准确解，但是可以求解最小二乘解。在 Matlab 中直接使用反斜线符便可以得到超定方程组的最小二乘解。它使用的是第三章提到的 QR 分解，注意，此时不需要显式计算 Q，只要得到矩阵 R 即可，然后回代即可得到最小二乘解。

参考文献 [1](P127-P129) 演示了一个人口预测的例子，建议使用原作者提供的 *qrsteps* 程序来运行一次这个例子，最后得到的上三角矩阵可以通过回代得到满足最小二乘的参数。下面是测试程序 (按下任意键进行下一次 QR 迭代):

code/testleastsquares.m

```
s=((1950:10:2000)'-1950)/50;
y=[150.697  179.323  203.212  226.505  249.633  281.422]';
X=[s.*s s ones(size(s))];
qrsteps(X,y)
ans =
  -1.251559027772961  -1.438206237226335  -1.757807623276631
                   0  -0.362715893232250  -1.301046138767852
                   0                   0   1.103354568734740
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
```

4.2　多项式插值

进行插值操作的多项式常见的有三种：标准形式 (standard form)，拉格朗日形式 (Lagrange form) 和牛顿形式 (Newton form)。标准形式就是常见的高次多项式形式，形如 $f(x) = a_n x^n + \ldots + a_1 x + a_0$，这个就不用介绍了。

4.2.1　拉格朗日插值多项式

拉格朗日插值多项式可以直接通过数据集得到多项式，不需要任何其他的计算。它的一般形式如下：

$$f(x) = \sum_{i=1}^{n} y_i L_i(x) = \sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

下面是它的 Matlab 代码实现：

code/lagrangeintpnm.m

```
function yint = lagrangeintpnm(x,y,z)
%使用拉格朗日插值多项式插值
% x=[1 2 4 5 7];
% y=[52 5 -5 -40 10];
% yint=lagrangeintpnm(x,y,3)
n=length(x);
for i=1:n
    L(i)=1;
    for j=1:n
        if j~= i
            L(i)=L(i)*((z-x(j))/(x(i)-x(j)));
        end
    end
end
yint=sum(y.*L);
end
```

拉格朗日插值多项式的优点是事先不需要进行任何计算便可以直接得到，但是对于任意一个 x，整个表达式便要重新计算一次，而且如果增加了新的数据点的话，所有的结果都要重新计算一次。Matlab 中函数 *polyinterp* 使用的便是拉格朗日插值多项式。它的源码和上面的代码略有不同，实质是一样的。

4.2.2 牛顿插值多项式

牛顿插值多项式可以适应不断增加的数据点，它之前计算结果不需要改变，只需要计算一次新增加的点的系数即可。它的一般形式如下：

$$f(x) = y_1 + f[x_2, x_1](x - x_1) + \ldots + f[x_n, x_{n-1}, \ldots, x_2, x_1](x - x_1)\ldots(x - x_{n1})$$

其中 $f[x_k, x_{k-1}\ldots, x_2, x_1]$ 表示差商 (divided difference)，它的计算式为：

$$f[x_k, x_{k-1}\ldots, x_2, x_1] = \frac{f[x_k, x_{k-1}\ldots, x_2] - f[x_{k-1}\ldots, x_2, x_1]}{x_k - x1}$$

例如：

$$a_1 = y_1 \quad a_2 = \frac{y_2 - y_1}{x_2 - x_1} \quad a_3 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1}$$

具体的计算方式可以参考文献 [3](P177-P181)。

下面是它的 Matlab 代码实现：

code/newtonintpnm.m

```
function yint = newtonintpnm(x,y,z)
%牛顿插值多项式插值
% x=[1 2 4 5 7];
% y=[52 5 -5 -40 10];
% yint=newtonintpnm(x,y,3)

n=length(x);
a(1)=y(1);
for i=1:(n-1)
    %第一列存放相邻的差商
    divdif(i,1)=(y(i+1)-y(i))/(x(i+1)-x(i));
end
for j=2:(n-1) %其他的列
    for i=1:(n-j) %其他列中的项数
        %计算其他的差商放到对应的列中，前面一列的下面一行与上面一行的差商
        divdif(i,j)=(divdif(i+1,j-1)-divdif(i,j-1))/(x(j+i)-x(i));
    end
end
for j=2:n
    a(j)=divdif(i,j-1);%参数的值是差商表中的第一行
end
yint=a(1);
xn=1;
for k=2:n
    xn=xn*(z-x(k-1));
    yint=yint+a(k)*xn;%计算插值
end
end
```

4.2.3 分段线性插值

分段插值就是插值多项式是分段的，常见的有分段线性插值，保形分段三次插值和三次样条插值。其中分段线性插值是最简单的，也是其他分段插值的基础。它是将数据点标记在坐标系中，然后用直线将这些点连接起来。很显然，这类插值结果的一阶导数是不连续的。Matlab 内置函数 *piecelie* 便是这类实现。后面不会介绍分段二次插值，如果需要了解可以参考文献 [3](P185-P187)。

code/piecelin.m

```
function v = piecelin(x,y,u)
%PIECELIN  Piecewise linear interpolation.
%  v = piecelin(x,y,u) finds the piecewise linear L(x)
%  with L(x(j)) = y(j) and returns v(k) = L(u(k)).

%  First divided difference

   delta = diff(y)./diff(x);

%  Find subinterval indices k so that x(k) <= u < x(k+1)

   n = length(x);
   k = ones(size(u));
   for j = 2:n-1
      k(x(j) <= u) = j;
   end

%  Evaluate interpolant

   s = u - x(k);
   v = y(k) + s.*delta(k);
```

4.2.4　保形分段三次插值

保形分段三次插值 (piecewise cubic Hermite interpolating polynomial) 是分段三次埃米特插值的一种，对于每个子区间都是使用三次多项式进行插值，它分了几种不同的情况对区间中点和左右端点处的斜率 d_k 进行讨论，详情可以参考文献 [1](P85-P86)。*pchip* 插值函数的一阶导数是连续的，但是二阶导数在节点处出现跳变。下面是原作者编写的 *pchip* 程序：

code/pchiptx.m

```
function v = pchiptx(x,y,u)
%PCHIPTX  Textbook piecewise cubic Hermite interpolation.
%  v = pchiptx(x,y,u) finds the shape-preserving piecewise cubic
%  interpolant P(x), with P(x(j)) = y(j), and returns v(k) = P(u(k))
    .
%
%  See PCHIP, SPLINETX.

%  First derivatives

   h = diff(x);
   delta = diff(y)./h;
   d = pchipslopes(h,delta);

%  Piecewise polynomial coefficients

   n = length(x);
   c = (3*delta - 2*d(1:n-1) - d(2:n))./h;
   b = (d(1:n-1) - 2*delta + d(2:n))./h.^2;

%  Find subinterval indices k so that x(k) <= u < x(k+1)

   k = ones(size(u));
   for j = 2:n-1
      k(x(j) <= u) = j;
```

```
    end

%  Evaluate interpolant

    s = u - x(k);
    v = y(k) + s.*(d(k) + s.*(c(k) + s.*b(k)));


%-------------------------------------------------------------

function d = pchipslopes(h,delta)
%  PCHIPSLOPES  Slopes for shape-preserving Hermite cubic
%  pchipslopes(h,delta) computes d(k) = P'(x(k)).

%  Slopes at interior points
%  delta = diff(y)./diff(x).
%  d(k) = 0 if delta(k-1) and delta(k) have opposites
%         signs or either is zero.
%  d(k) = weighted harmonic mean of delta(k-1) and
%         delta(k) if they have the same sign.

    n = length(h)+1;
    d = zeros(size(h));
    k = find(sign(delta(1:n-2)).*sign(delta(2:n-1))>0)+1;
    w1 = 2*h(k)+h(k-1);
    w2 = h(k)+2*h(k-1);
    d(k) = (w1+w2)./(w1./delta(k-1) + w2./delta(k));

%  Slopes at endpoints

    d(1) = pchipend(h(1),h(2),delta(1),delta(2));
    d(n) = pchipend(h(n-1),h(n-2),delta(n-1),delta(n-2));

%-------------------------------------------------------------
```

```
function d = pchipend(h1,h2,del1,del2)
%  Noncentered, shape-preserving, three-point formula.
   d = ((2*h1+h2)*del1 - h1*del2)/(h1+h2);
   if sign(d) ~= sign(del1)
      d = 0;
   elseif (sign(del1)~=sign(del2))&(abs(d)>abs(3*del1))
      d = 3*del1;
   end
```

4.2.5　三次样条插值

三次样条插值 (cubic spline) 是另一种分段三次埃米特插值，它的插值函数的一阶导数和二阶导数都是连续的。

(1) 首先将原区间分成 $n-1$ 个子区间 (区间长度可以不相同)，每个子区间的左右端点都要满足该区间的三次多项式，那么就可以得到 $2(n-1)=(2n-2)$ 个方程;

(2) 然后对各个子区间的三次多项式进行求导得到的导函数要保证相邻两个子区间的交点处的导函数值相等，那么就可以得到 $(n-2)$ 个方程;

(3) 接着为了保证二阶导数也是连续的，类似上面的操作对每个子区间的三次多项式求二阶导数然后使得子区间交点处导函数值相同，那么又可以得到 $(n-2)$ 个方程;

(4) 对于 (n-1) 个三次多项式共有 $(4n-4)$ 个参数，需要 $(4n-4)$ 个方程，现在有了 $(4n-6)$ 个方程，还差两个，这时候可以自行给定附加条件。例如：令整个区间的左右端点的二阶导数为 0，这个便是自然三次样条插值 (natural cubic splines)。但是 Matlab 内置函数使用的是“not-a-knot”(非节点) 这个条件，它是指在最开始和最后的两个子区间 $x_1 \le x \le x_3$ 和 $x_{n-2} \le x \le x_n$ 上分别使用一个单独的三次多项式，就相当于认为 x_2 和 x_{n-1} 都不是节点。

该算法还是比较复杂的，下面是原作者编写的 *spline* 程序：

code/splinetx.m

```
function v = splinetx(x,y,u)
%SPLINETX  Textbook spline function.
%  v = splinetx(x,y,u) finds the piecewise cubic interpolatory
%  spline S(x), with S(x(j)) = y(j), and returns v(k) = S(u(k)).
%
%  See SPLINE, PCHIPTX.

%  First derivatives

   h = diff(x);
   delta = diff(y)./h;
   d = splineslopes(h,delta);

%  Piecewise polynomial coefficients

   n = length(x);
   c = (3*delta - 2*d(1:n-1) - d(2:n))./h;
   b = (d(1:n-1) - 2*delta + d(2:n))./h.^2;

%  Find subinterval indices k so that x(k) <= u < x(k+1)

   k = ones(size(u));
   for j = 2:n-1
      k(x(j) <= u) = j;
   end

%  Evaluate spline

   s = u - x(k);
   v = y(k) + s.*(d(k) + s.*(c(k) + s.*b(k)));


%-------------------------------------------------------------------
```

```
function d = splineslopes(h,delta)
%  SPLINESLOPES  Slopes for cubic spline interpolation.
%  splineslopes(h,delta) computes d(k) = S'(x(k)).
%  Uses not-a-knot end conditions.

%  Diagonals of tridiagonal system

   n = length(h)+1;
   a = zeros(size(h)); b = a; c = a; r = a;
   a(1:n-2) = h(2:n-1);
   a(n-1) = h(n-2)+h(n-1);
   b(1) = h(2);
   b(2:n-1) = 2*(h(2:n-1)+h(1:n-2));
   b(n) = h(n-2);
   c(1) = h(1)+h(2);
   c(2:n-1) = h(1:n-2);

%  Right-hand side

   r(1) = ((h(1)+2*c(1))*h(2)*delta(1)+ ...
           h(1)^2*delta(2))/c(1);
   r(2:n-1) = 3*(h(2:n-1).*delta(1:n-2)+ ...
               h(1:n-2).*delta(2:n-1));
   r(n) = (h(n-1)^2*delta(n-2)+ ...
           (2*a(n-1)+h(n-1))*h(n-2)*delta(n-1))/a(n-1);

%  Solve tridiagonal linear system

   d = tridisolve(a,b,c,r);
```

4.3 Matlab 函数解析

Matlab 内置的曲线拟合和插值函数

(1) 曲线拟合函数 $polyfit$

$p = polyfit(x, y, m)$

(2) 多项式插值函数 $polyinterp$ 使用拉格朗日插值多项式

$v = polyinerp(x, y, u)$

(3) 插值函数 $interp1$ (最后一个字母是数值“1”)

$yi = interp1(x, y, xi, 'method')$

向量 x 中的元素必须是单调的，可选的方法包括：

'nearest' 返回最接近插值点的数据点的值

'linear' 使用分段线性插值

'spline' 使用三次样条插值

'pchip' 使用三次埃米特插值

'cubic' 等同于'pchip'

对于'nearest' 和'linear' 方法只能估计定义域 x 内的插值点的函数值，然后方法'spline' 和'pchip' 可以估计定义域 x 外的插值点的函数值

参考文献

[1] Numerical Computing with Matlab. Cleve B. Moler.
中文翻译版本《Matlab 数值计算》，喻文健，机械工业出版社，2006，6
网站资源：
(1)Cleve Moler 撰写的教科书
(2)数值计算交互演示网站

[2] 数值分析与算法，喻文健，清华大学出版社，2012，1

[3] Numerical Methods: An introduction with Applications Using Matlab. Amos Gilat. Vish Subramaniam，2010，10

[4] Data Mining Algorithms In R.
网站资源:
WikiBook：Data Mining Algorithms In R